Documento accompagnatorio al progetto

Istruzioni per la compilazione e l'esecuzione

Per lo svilupo del progetto sono stati usati i seguenti programmi:

Programma	Versione
`make`	`GNU Make 4.2.1`
`gcc`	`9.4.0 (Ubuntu)`
`flex`	`2.6.4`
`bison`	`GNU Bison 3.5.1`

Per la compilazione è possibile usare il comando `make` o, in alternativa, i tre comandi:

```
# 1. Compila il file di flex
flex -o bison-flex/lex.yy.c bison-flex/digraph.l

# 2. Compila il file di bison
bison --defines=bison-flex/digraph.tab.h bison-flex/digraph.y \
    -o bison-flex/digraph.tab.c

# 3. Compila tutti i sorgenti e produce l'eseguibile
gcc bison-flex/lex.yy.c bison-flex/digraph.tab.c \
  src/list/list.c src/digraph/digraph.c src/map/map.c \
  src/subset_construction/subset_construction.c \
  src/saver/saver.c src/program.c \
    -o ./parser
```

Per eseguire il programma usare `./parser` fornendo il percorso a un file di testo contenente la rappresentazione in dot notation di un grafo. Il DFA ottenuto viene scritto nel file `out.dot`.

Il programma in poche parole

Contenstualmente al parsing viene creato il grafo rappresentato dal file di input, controllando che gli attributi dichiarati siano tra quelli accettati e che tutti i nodi abbiano un `id` non vuoto. Fatto ciò, il grafo "grezzo" così ottenuto viene controllato per verificare che:

(a) Il grafo abbia esattamente un punto d'ingresso (un nodo con `id="0"`);

(b) Il grafo abbia almeno un punto d'accettazione (un nodo con `shape="doublecircle"`);

(c) Tutti gli archi coinvolgano nodi presenti nel grafo;

NB. I valori degli attributi sono valutati in modo *case-insensitive*, mentre gli id dei nodi sono *case-sensitive*.

Terminati questi controlli, il grafo ottenuto viene passato alla funzione che implementa l'algoritmo di *Subset construction*. La funzione rappresenta i sottoinsiemi di nodi usando una struttura `STATE` che, tra le altre cose, contiene una lista di puntatori ai nodi che comprende. Al termine, viene creato un nuovo grafo in cui ciascun nodo è ottenuto da uno degli `STATE` individuati.

Il grafo così ottenuto viene tradotto in *DOT notation* e salvato sul file di output.

Note sull'implementazione della hashmap

L'implementazione della hashmap (file `map.h` e `map.c`) è stata presa da qui questa repository GitHub: (`https://github.com/Mashpoe/c-hashmap`).